IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In the matter of:

Appl. No.	:	09/934,156	Confirmation No. 7387
Applicant	:	David Roth Rigney	
Filed	:	August 21, 2001	
TC/A.U.	:	2168	
Examiner	:	Cheyne D. Ly	
Response to	:	Office Action with mailing date of April 20, 2006	

DECLARATION OF DAVID R. RIGNEY
WITH AN ATTACHED DOCUMENT ENTITLED "rainbow --help"

I, David R. Rigney, declare as follows:

1. I am the inventor named in the patent application referenced above. I make this Declaration based on my personal knowledge, and could and would testify competently to the facts stated herein.

2. On June 27, 2006, I created two compact discs (named COPY 1 and COPY 2) that are submitted herewith, along with a transmittal letter that describes the contents and usage of the software that are contained on the compact discs. The COPY 2 compact disc is a duplicate of the COPY 1 compact disc. I scanned the discs with ClamWin Antivirus software (Engine version 0.88.3, freely available from www.clamwin.com) and found that there were no virus-infected files on the discs.

3. Both of the compact discs contain computer listings of freely available, public domain software: the November 22, 1999 version of the "bow" software toolkit ("bow-19991122"). I downloaded the "bow-19991122" software toolkit on March 18, 2000 from the Web site www.cs.cmu.edu/~mccallum/bow in compressed form, then uncompressed and extracted it (with the UNIX "tar" utility program). The toolkit is contained on the compact disc within a directory named "bow-19991122", which contains 109 files (total size 1,868,349 bytes) and two subdirectories. The subdirectories are "argp", containing 32 files (total size 274,860 bytes), and "bow", containing 17 files (total size 128,038 bytes). The names of the individual files within the

main-directory and sub-directories of the toolkit, along with their size in bytes and times & dates of creation, are listed in an attachment to the transmittal letter that accompanies the compact discs.

4. The "bow" software toolkit is intended to be installed and run in a UNIX operating system environment, e.g., Linux , or DJGPP within a MS-Windows system. A preliminary to installing the "bow" software library is described in the file "E:\bow-19991122\HACKING" on the compact disc, namely, to run the "autoconf" program. Then, compiling and installing the "bow" software library are performed as described in the file on the compact disc "E:\bow-19991122\INSTALL", and sample programs may then be run (e.g., the "Rainbow" program.). I performed all these tasks on March 22, 2000, within a UNIX operating system environment, namely, DJGPP running within a MS-Windows system.

5. The program "Rainbow", which accompanies the "bow" software toolkit, serves as a front-end for the toolkit, allowing its user to make use of the functions of the toolkit through command-line commands that the user provides to Rainbow. The file "E:\Rainbow.htm" on each compact disc is a tutorial describing usage of the computer program "Rainbow", last updated by its author, Andrew McCallum, on September 30, 1998. The tutorial is readable by a Web browser such as Microsoft's Internet Explorer. I copied that tutorial file from the Web site www.cs.cmu.edu/~mccallum/bow on March 22, 2000.

6. As described in the above-mentioned tutorial:

> You can obtain on-line documentation of each rainbow command-line option by typing
> rainbow --help | more

The online documentation that I obtained after typing the "rainbow --help" command is printed out in an appendix to this Declaration. The appendix is thirteen pages long and is entitled "rainbow --help". It is a true and exact copy of the output of the "rainbow" computer program in response to the command "rainbow --help".

7. The above-mentioned tutorial is also useful for testing the operation of "Rainbow" once it has been installed as described above in paragraph 4. As described in the tutorial:

> The examples on this page assume that you have compiled libbow [the bow toolkit library] and rainbow, and that rainbow is in your path. Several of the examples also assume that you have downloaded the 20_newsgroups data set, unpacked it in your home directory, and therefore that its files are available in the directory ~/20_newsgroups. ...
>
> The general pattern of rainbow usage is in two steps (1) have rainbow read your documents and write to disk a "model" containing their statistics, (2) using the model, rainbow performs classification or diagnostics.

The "20_newsgroups data set," referred to above, is a set of 20 collections of text files, each of which is itself a collection of text files pertaining to a particular subject, such as baseball, motorcycles, medicine, or mideast politics. More specifically, it is a collection of UseNet postings from 20 newsgroups, gathered by Ken Lang at Carnegie Mellon University in the mid-1990s. On March 22, 2000, I downloaded the 20_newsgroups data set from the Web site www.cs.cmu.edu/~mccallum/bow, and I unpacked it as instructed in the tutorial. (This same data set was described in my Declaration and Amendments dated July 7, 2004 that were made in response to the Office Action with a mailing date of January 8, 2004.) I then tested and learned usage of the "Rainbow" computer program, by following the instructions and examples that are provided in the "Rainbow.htm" tutorial that is included on compact discs that are submitted herewith. I obtained additional experience in the usage of the "Rainbow" computer program by executing the commands described in its on-line help, which are printed out in the appendix to this Declaration, as suggested in the "Rainbow.htm" tutorial (see paragraph 6 above). Finally, I consulted readily available introductory literature such as references No. 21 (MANNING and SCHUTZE, 1999) and No. 22 (MITCHELL, 1997) of the Information Disclosure for this application, in order to understand the rationale of the algorithms that are implemented by the "bow" software toolkit. For example, MITCHELL (1997) has a discussion of the default classification method (Naive Bayes), including its Table 6.2 that shows the Naive Bayes algorithm itself and its Table 6.3 that describes the above-mentioned 20_newsgroups data set.

8. The amendments to the instant application, which are submitted herewith, contain incorporation by reference of the same published material that is contained in the compact disc described in this Declaration. That material was well-known and readily available to the public at the time of the original filing of the instant application. One of ordinary skill in the art, given my original disclosure, would clearly see that I had possession of the material contained on that compact disc, as well as any other materials described in this Declaration, at the time of original filing. This is because the "bow-19991122" software toolkit that is included on the compact disc is specifically referred to on page 17, lines 5-12 of the application, including reference to the Web site from which it was downloaded, reference to its open source code, and reference to its installation as described in paragraph 4 above (including the need to run the "autoconf" program as a preliminary). Furthermore, the actual usage and functioning of the "rainbow" program, as described in the tutorial "Rainbow.htm" (which is the cited in the application as McCALLUM (1998)) is specifically referred to in the application on page 18, lines 9-15, and additional such specific references are made throughout the application, e.g., page 20, lines 16-22; and page 37, line 3 through page 42, line12. The appendix to this Declaration is also specifically referred to on page 37, lines 18-21 of the original application, as the "rainbow --help" on-line documentation.

9. One of ordinary skill in the art, given my original disclosure, given the published material that is contained in the compact disc "bow-19991122", and given access to (and rudimentary familiarity with) a UNIX-based software environment, would be able to install and operate the "rainbow" computer program; and would be able to understand the function of the "rainbow" computer program in the general context of text data-sets (after having downloaded the "20_newsgroups data set", after having worked through the examples in the "Rainbow.htm" tutorial, and after having expanded on those examples by executing additional commands listed in the "rainbow--help" appendix.). One of ordinary skill in the art at the time of my original disclosure, given published literature such as references No. 21 (MANNING and SCHUTZE, 1999) and No. 22 (MITCHELL, 1997) of the Information Disclosure for this application, would also be able to understand the rationale for algorithms that are implemented as methods of the "rainbow" computer program, such as the Naive Bayes method.

I declare under penalty of perjury under the laws of the United States of America that all the foregoing is true and correct.

Signed in Austin, Texas on July 18, 2006:

David R Rigney

David R. Rigney

GENETWORKS Inc.
P.O. Box 33296
Austin TX 78764-0296
Tel. 512-445-7301
drigney@genetworks.com

CERTIFICATE OF EXPRESS MAIL UNDER 37 C.F.R. 1.10

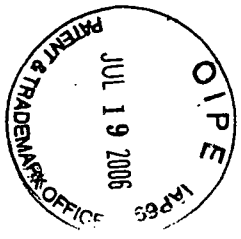
I hereby certify that this paper is being deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated below and is addressed to Mail Stop RCE, Commissioner for Patents, P.O. Box 1450, Alexandria VA 22313-1450.

Printed Name: David R. Rigney

Date of Deposit: July 19, 2006

Signature: David R Rigney

Express Mail Label No. EQ 833932995 US




```
Usage: rainbow.exe [OPTION...] [ARG...]
Rainbow -- a document classification front-end to libbow

 Testing documents that are specified on the command line:
  -x, --test-files            In same format as `-t', output classifications of
                              documents in the directory ARG  The ARG must have
                              the same subdir names as the ARG's specified when
                              --index'ing.
  -X, --test-files-loo        Same as --test-files, but evaulate the files
                              assuming that they were part of the training data,
                              and doing leave-one-out cross-validation.  This
                              only works with the classification methods that
                              support leave-one-out evaluation

 Splitting options:
      --ignore-set=SOURCE     How to select the ignored documents.  Same format
                              as --test-set.  Default is `0'.
      --set-files-use-basename[=N]
                              When using files to specify doc types, compare
                              only the last N components the doc's pathname.
                              That is use the filename and the last N-1
                              directory names.  If N is not specified, it
                              defaults to 1.
      --test-set=SOURCE       How to select the testing documents.  A number
                              between 0 and 1 inclusive with a decimal point
                              indicates a random fraction of all documents.  The
                              number of documents selected from each class is
                              determined by attempting to match the proportions
                              of the non-ignore documents.  A number with no
                              decimal point indicates the number of documents to
                              select randomly.  Alternatively, a suffix of `pc'
                              indicates the number of documents per-class to
                              tag.  `remaining' selects all documents that
                              remain untagged at the end.  Anything else is
                              interpreted as a filename listing documents to
                              select.  Default is `0.3'.
      --train-set=SOURCE      How to select the training documents.  Same format
                              as --test-set.  Default is `remaining'.
      --unlabeled-set=SOURCE How to select the unlabeled documents.  Same
                              format as --test-set.  Default is `0'.
```

```
     --validation-set=SOURCE   How to select the validation documents.  Same
                               format as --test-set.  Default is `0'.

For building data structures from text files:
     --index-matrix=FORMAT  Read document/word statistics from a file in the
                            format produced by --print-matrix=FORMAT.  See
                            --print-matrix for details about FORMAT.
 -i, --index                Tokenize training documents found under
                            directories ARG... (where each ARG directory
                            contains documents of a different class), build
                            token-document matrix, and save it to disk.

For doing document classification using the token-document matrix built with
-i:
 -q, --query[=FILE]         Tokenize input from stdin [or FILE], then print
                            classification scores.
     --query-server=PORTNUM Run rainbow in server mode, listening on socket
                            number PORTNUM.  You can try it by executing this
                            command, then in a different shell window on the
                            same machine typing `telnet localhost PORTNUM'.
 -r, --repeat               Prompt for repeated queries.

Rainbow-specific vocabulary options:
     --hide-vocab-in-file=FILE   Hide from the vocabulary all words read as
                            space-separated strings from FILE.  Note that
                            regular lexing is not done on these strings.
     --hide-vocab-indices-in-file=FILE
                            Hide from the vocabulary all words read as
                            space-separated word integer indices from FILE.
     --use-vocab-in-file=FILE   Limit vocabulary to just those words read as
                            space-separated strings from FILE.  Note that
                            regular lexing is not done on these strings.

Testing documents that were indexed with `-i':
 -t, --test=N               Perform N test/train splits of the indexed
                            documents, and output classifications of all test
                            documents each time.  The parameters of the
                            test/train splits are determined by the option
                            `--test-set' and its siblings
     --test-on-training=N   Like `--test', but instead of classifing the
```

```
                          held-out test documents classify the training data
                          in leave-one-out fashion.  Perform N trials.

Diagnostics:
     --build-and-save     Builds a class model and saves it to disk.  This
                          option is unstable.
 -B, --print-matrix[=FORMAT]   Print the word/document count matrix in an awk-
                          or perl-accessible format.  Format is specified by
                          the following letters:
                          print all vocab or just words in document:
                            a=all OR s=sparse
                          print counts as ints or binary:
                            b=binary OR i=integer
                          print word as:
                              n=integer index OR w=string OR e=empty OR
                          c=combination
                          The default is the last in each list
 -F, --print-word-foilgain=CLASSNAME
                          Print the word/foilgain vector for CLASSNAME.  See
                          Mitchell's Machine Learning textbook for a
                          description of foilgain.
 -I, --print-word-infogain=N   Print the N words with the highest information
                          gain.
     --print-doc-names[=TAG]   Print the filenames of documents contained in
                          the model.  If the optional TAG argument is given,
                          print only the documents that have the specified
                          tag, where TAG might be `train', `test', etc.
     --print-log-odds-ratio[=N]   For each class, print the N words with the
                          highest log odds ratio score.  Default is N=10.
     --print-word-counts=WORD   Print the number of times WORD occurs in each
                          class.
     --print-word-pair-infogain=N
                          Print the N word-pairs, which when co-occuring in
                          a document, have the highest information gain.
                          (Unfinished; ignores N.)
     --print-word-probabilities=CLASS
                          Print P(w|CLASS), the probability in class CLASS
                          of each word in the vocabulary.
     --test-from-saved    Classify using the class model saved to disk.
                          This option is unstable.
```

```
 -W, --print-word-weights=CLASSNAME
                            Print the word/weight vector for CLASSNAME, sorted
                            with high weights first.  The meaning of `weight'
                            is undefined.

Probabilistic Indexing options, --method=prind:
 -G, --prind-no-foilgain-weight-scaling
                            Don't have PrInd scale its weights by Quinlan's
                            FoilGain.
 -N, --prind-no-score-normalization
                            Don't have PrInd normalize its class scores to sum
                            to one.
     --prind-non-uniform-priors   Make PrInd use non-uniform class priors.

General options
     --annotations=FILE     The sarray file containing annotations for the
                            files in the index
 -b, --no-backspaces        Don't use backspace when verbosifying progress
                            (good for use in emacs)
 -d, --data-dir=DIR         Set the directory in which to read/write
                            word-vector data (default=~/.<program_name>).
     --random-seed=NUM      The non-negative integer to use for seeding the
                            random number generator
     --score-precision=NUM  The number of decimal digits to print when
                            displaying document scores
 -v, --verbosity=LEVEL      Set amount of info printed while running;
                            (0=silent, 1=quiet, 2=show-progess,...5=max)

Lexing options
     --append-stoplist-file=FILE
                            Add words in FILE to the stoplist.
     --exclude-filename=FILENAME
                            When scanning directories for text files, skip
                            files with name matching FILENAME.
 -g, --gram-size=N          Create tokens for all 1-grams,... N-grams.
 -h, --skip-header          Avoid lexing news/mail headers by scanning forward
                            until two newlines.
     --istext-avoid-uuencode   Check for uuencoded blocks before saying that
                            the file is text, and say no if there are many
                            lines of the same length.
```

```
      --lex-pipe-command=SHELLCMD
                            Pipe files through this shell command before
                            lexing them.
      --no-stemming         Do not modify lexed words with a stemming
                            function. (usually the default, depending on
                            lexer)
      --replace-stoplist-file=FILE
                            Empty the default stoplist, and add
                            space-delimited words from FILE.
  -s, --no-stoplist         Do not toss lexed words that appear in the
                            stoplist.
      --shortest-word=LENGTH Toss lexed words that are shorter than LENGTH.
                            Default is usually 2.
  -S, --use-stemming        Modify lexed words with the `Porter' stemming
                            function.
      --use-stoplist        Toss lexed words that appear in the stoplist.
                            (usually the default SMART stoplist, depending on
                            lexer)

Mutually exclusive choice of lexers
      --flex-mail           Use a mail-specific flex lexer
      --flex-tagged         Use a tagged flex lexer
  -H, --skip-html           Skip HTML tokens when lexing.
      --lex-alphanum        Use a special lexer that includes digits in
                            tokens, delimiting tokens only by non-alphanumeric
                            characters.
      --lex-suffixing       Use a special lexer that adds suffixes depending
                            on Email-style headers.
      --lex-white           Use a special lexer that delimits tokens by
                            whitespace only, and does not change the contents
                            of the token at all---no downcasing, no stemming,
                            no stoplist, nothing.  Ideal for use with an
                            externally-written lexer interfaced to rainbow
                            with --lex-pipe-cmd.

Feature-selection options
  -D, --prune-vocab-by-doc-count=N
                            Remove words that occur in N or fewer documents.
  -O, --prune-vocab-by-occur-count=N
                            Remove words that occur less than N times.
```

```
 -T, --prune-vocab-by-infogain=N
                            Remove all but the top N words by selecting words
                            with highest information gain.

Weight-vector setting/scoring method options
     --binary-word-counts   Instead of using integer occurrence counts of
                            words to set weights, use binary absence/presence.

     --event-document-then-word-document-length=NUM
                            Set the normalized length of documents when
                            --event-model=document-then-word
     --event-model=EVENTNAME   Set what objects will be considered the
                            `events' of the probabilistic model.  EVENTNAME
                            can be one of: word, document, document-then-word.
                             Default is `word'.
     --infogain-event-model=EVENTNAME
                            Set what objects will be considered the `events'
                            when information gain is calculated.  EVENTNAME
                            can be one of: word, document, document-then-word.
                             Default is `document'.
 -m, --method=METHOD        Set the word weight-setting method; METHOD may be
                            one of: svm, tfidf, tfidf_log_occur,
                            tfidf_log_words, tfidf_words, prind, nbsimple,
                            nbshrinkage, naivebayes, maxent, knn, kl,
                            emsimple, em, dirk, active, default=naivebayes.
     --print-word-scores    During scoring, print the contribution of each
                            word to each class.
     --smoothing-goodturing-k=NUM
                            Smooth word probabilities for words that occur NUM
                            or less times. The default is 7.
     --smoothing-method=METHOD   Set the method for smoothing word
                            probabilities to avoid zeros; METHOD may be one
                            of: goodturing, laplace, mestimate, wittenbell
     --uniform-class-priors When setting weights, calculating infogain and
                            scoring, use equal prior probabilities on classes.


Active Learning options:
     --active-add-per-round=NUM   Specify the number of documents to label
                            each round.  The default is 4.
```

```
      --active-beta=NUM       Increase spread of document densities.
      --active-binary-pos=CLASS   The name of the positive class for binary
                              classification.  Required forrelevance sampling.
      --active-committee-size=NUM
                              The number of committee members to use with QBC.
                              Default is 1.
      --active-final-em       Finish with a full round of EM.
      --active-no-final-em    Finish without a full round of EM.
      --active-num-rounds=NUM   The number of active learning rounds to
                              perform.  The default is 10.
      --active-perturb-after-em   Perturb after running EM to create committee
                              members.
      --active-pr-print-stat-summary
                              Print the precision recall curves used for score
                              to probability remapping.
      --active-pr-window-size=NUM
                              Set the window size for precision-recall score to
                              probability remapping.The default is 20.
      --active-print-committee-matrices
                              Print the confusion matrix for each committee
                              member at each round.
      --active-qbc-low-kl     Select documents with the lowest kl-divergence
                              instead of the highest.
      --active-remap-scores-pr   Remap scores with sneaky precision-recall
                              tricks.
      --active-secondary-method=METHOD
                              The underlying method for active learning to use.
                              The default is 'naivebayes'.
      --active-selection-method=METHOD
                              Specify the selection method for picking unlabeled
                              docs. One of uncertainty, relevance, qbc, random.
                              The default is 'uncertainty'.
      --active-stream-epsilon=NUM
                              The rate factor for selecting documents in stream
                              sampling.
      --active-test-stats     Generate output for test docs every n rounds.

Dirichlet Kernel options, --method=dirk:
      --dirk-print-alphas     Print the alphas of the Beta distribution learned
                              for each word of the learned prior.
```

```
     --dirk-prior-alpha=NUM Set the prior alpha parameter.  Defaults to 1.0.

EM options:
     --em-anneal            Use Deterministic annealing EM.
     --em-anneal-normalizer When running EM, do deterministic annealing-ish
                            stuff with the unlabeled normalizer.
     --em-binary            Do special tricks for the binary case.
     --em-binary-neg-classname=CLASS
                            Specify the name of the negative class if building
                            a binary classifier.
     --em-binary-pos-classname=CLASS
                            Specify the name of the positive class if building
                            a binary classifier.
     --em-compare-to-nb     When building an EM class barrel, show doc stats
                            for the naivebayesbarrel equivalent.  Only use in
                            conjunction with --test.
     --em-crossentropy      Use crossentropy instead of naivebayes for
                            scoring.
     --em-halt-using-accuracy=TYPE
                            When running EM, halt when accuracy plateaus.
                            TYPE is type of document to measure perplexity on.
                             Choices are `validation', `train', `test',
                            `unlabeled' and `trainandunlabeled' and
                            `trainandunlabeledloo'
     --em-halt-using-perplexity=TYPE
                            When running EM, halt when perplexity plataeus.
                            TYPE is type of document to measure perplexity on.
                             Choices are `validation', `train', `test',
                            `unlabeled',  `trainandunlabeled' and
                            `trainandunlabeledloo'
     --em-multi-hump-init=METHOD
                            When initializing mixture components, how to
                            assign component probs to documents.  Default is
                            `spread'.  Other choices are `spiked'.
     --em-multi-hump-neg=NUM   Use NUM center negative classes. Only use in
                            binary case.Must be using scoring method
                            nb_score.
     --em-num-iterations=NUM   Number of EM iterations to run when building
                            model.
     --em-perturb-starting-point=TYPE
```

```
                          Instead of starting EM with P(w|c) from the
                          labeled training data, start from values that are
                          randomly sampled from the multinomial specified by
                          the labeled training data.  TYPE specifies what
                          distribution to use for the perturbation; choices
                          are `gaussian' `dirichlet', and `none'.  Default
                          is `none'.
     --em-print-accuracy=TYPE   When running EM, print the accuracy of
                          documents at each round.  TYPE is type of document
                          to measure perplexity on.  See
                          `--em-halt-using-perplexity` for choices for TYPE
     --em-print-perplexity=TYPE   When running EM, print the perplexity of
                          documents at each round.  TYPE is type of document
                          to measure perplexity on.  See
                          `--em-halt-using-perplexity` for choices for TYPE
     --em-print-top-words    Print the top 10 words per class for each EM
                          iteration.
     --em-save-probs         On each EM iteration, save all P(C|w) to a file.
     --em-stat-method=STAT   The method to convert scores to probabilities.The
                          default is 'nb_score'.
     --em-temp-reduce=NUM    Temperature reduction factor for deterministic
                          annealing.  Default is 0.9.
     --em-temperature=NUM    Initial temperature for deterministic annealing.
                          Default is 200.
     --em-unlabeled-normalizer=NUM
                          Number of unlabeled docs it takes to equal a
                          labeled doc.Defaults to one.
     --em-unlabeled-start=TYPE   When initializing the EM starting point, how
                          the unlabeled docs contribute.  Default is `zero'.
                           Other choices are `prior' `random'  and `even'.

EMSIMPLE options:
     --emsimple-num-iterations=NUM
                          Number of EM iterations to run when building
                          model.
     --emsimple-print-accuracy=TYPE
                          When running emsimple, print the accuracy of
                          documents at each EM round.  Type can be
                          validation, train, or test.
```

```
K-nearest neighbor options, --method=knn:
     --knn-k=K                Number of neighbours to use for nearest neighbour.
                              Defaults to 30.
     --knn-weighting=xxx.xxx  Weighting scheme to use, coded like SMART.
                              Defaults to nnn.nnnThe first three chars describe
                              how the model documents areweighted, the second
                              three describe how the test document isweighted.
                              The codes for each position are described in
                              knn.c.Classification consists of summing the
                              scores per class for thek nearest neighbour
                              documents and sorting.

Maximum Entropy options, --method=maxent:
     --maxent-gaussian-prior  Add a Gaussian prior to each word/class feature
                              constraint.
     --maxent-gaussian-prior-no-zero-constraints
                              When using a gaussian prior, do not enforce
                              constraints that have notraining data.
     --maxent-halt-by-accuracy=TYPE
                              When running maxent, halt iterations using the
                              accuracy of documents.  TYPE is type of
                              documentsto test.  See
                              `--em-halt-using-perplexity` for choices for TYPE
     --maxent-halt-by-logprob=TYPE
                              When running maxent, halt iterations using the
                              logprob of documents.  TYPE is type of documentsto
                              test.  See `--em-halt-using-perplexity` for
                              choices for TYPE
     --maxent-iterations=NUM  The number of iterative scaling iterations to
                              perform.  The default is 40.
     --maxent-keep-features-by-mi=NUM
                              The number of top words by mutual information per
                              class to use as features.  Zeroimplies no pruning
                              and is the default.
     --maxent-logprob-constraints
                              Set constraints to be the log prob of the word.
     --maxent-print-accuracy=TYPE
                              When running maximum entropy, print the accuracy
                              of documents at each round.  TYPE is type of
                              document to measure perplexity on.  See
```

```
                          `--em-halt-using-perplexity` for choices for TYPE
      --maxent-prior-variance=NUM
                          The variance to use for the Gaussian prior.  The
                          default is 0.01.
      --maxent-prune-features-by-count=NUM
                          Prune the word/class feature set, keeping only
                          those features that haveat least NUM occurrences
                          in the training set.
      --maxent-scoring-hack  Use smoothed naive Bayes probability for zero
                          occuring word/class pairs during scoring
      --maxent-smooth-counts Add 1 to the count of each word/class pair when
                          calculating the constraint values.
      --maxent-vary-prior-by-count
                          Multiply log (1 + N(w,c)) times variance when
                          using a gaussian prior.
      --maxent-vary-prior-by-count-linearly
                          Mulitple N(w,c) times variance when using a
                          Gaussian prior.

Naive Bayes options, --method=naivebayes:
      --naivebayes-binary-scoring
                          When using naivebayes, use hacky scoring to get
                          good Precision-Recall curves.
      --naivebayes-m-est-m=M When using `m'-estimates for smoothing in
                          NaiveBayes, use M as the value for `m'.  The
                          default is the size of vocabulary.
      --naivebayes-normalize-log   When using naivebayes, return -1/log(P(C|d),
                          normalized to sum to one instead of P(C|d).  This
                          results in values that are not so close to zero
                          and one.

Support Vector Machine options, --method=svm:
      --svm-active-learning= Use active learning to query the labels &
                          incrementally (by arg_size) build the barrels.
      --svm-active-learning-baseline=
                          Incrementally add documents to the training set at
                          random.
      --svm-al_init_tsetsize=   Number of random documents to start with in
                          active learning.
      --svm-bsize=        maximum size to construct the subproblems.
```

```
      --svm-cache-size=      Number of kernel evaluations to cache.
      --svm-cost=            cost to bound the lagrange multipliers by (default
                             1000).
      --svm-df-counts=       Set df_counts (0=occurrences, 1=words).
      --svm-epsilon_a=       tolerance for the bounds of the lagrange
                             multipliers (default 0.0001).
      --svm-kernel=          type of kernel to use (0=linear, 1=polynomial,
                             2=gassian, 3=sigmoid, 4=fisher kernel).
      --svm-quick-scoring    Turn quick scoring on.
      --svm-remove-misclassified   Remove all of the misclassified examples and
                             retrain.
      --svm-rseed=           what random seed should be used in the
                             test-in-train splits
      --svm-start-at=        which model should be the first generated.
      --svm-suppress-score-matrix
                             Do not print the scores of each test document at
                             each AL iteration.
      --svm-test-in-train    do active learning testing inside of the
                             training...  a hack around making code 10 times
                             more complicated.
      --svm-tf-transform=    0=raw, 1=log...
      --svm-trans-cost=      value to assign to C* (default 200).
      --svm-trans-npos=      number of unlabeled documents to label as positive
                             (default: proportional to number of labeled
                             positive docs).
      --svm-transduce-class= override default class(es) (int) to do
                             transduction with (default bow_doc_unlabeled).
      --svm-use-smo=         default 1 (use SMO) - PR_LOQO not compiled
      --svm-vote=            Type of voting to use (0=singular, 1=pairwise;
                             default 0).
      --svm-weight=          type of function to use to set the weights of the
                             documents' words (0=raw_frequency, 1=tfidf,
                             2=infogain.

  -?, --help                 Give this help list
      --usage                Give a short usage message
  -V, --version              Print program version

Mandatory or optional arguments to long options are also mandatory or optional
for any corresponding short options.
```

Report bugs to <mccallum@cs.cmu.edu>.